MaeSTrO: A Mobile App for Style Transfer Orchestration using Neural Networks*

Max Reimann
Hasso Plattner Institute
University of Potsdam

Mandy Klingbeil, Sebastian Pasewaldt
Digital Masterpieces GmbH

Amir Semmo, Matthias Trapp, Jürgen Döllner
Hasso Plattner Institute
University of Potsdam

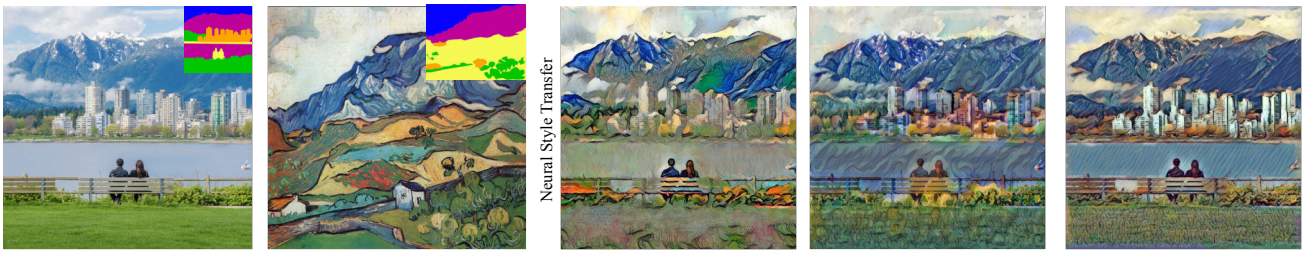


Fig. 1: Comparison of neural style transfer techniques implemented in *MaeSTrO*. Interactive location-based filtering is used to stylize the content image, using extensions to global style transfer techniques to provide more expressive results. Content image © karamysh on shutterstock.com, used with permission. The style image by Vincent van Gogh is in the public domain.

***Abstract*—Mobile expressive rendering gained increasing popularity among users seeking casual creativity by image stylization and supports the development of mobile artists as a new user group. In particular, neural style transfer has advanced as a core technology to emulate characteristics of manifold artistic styles. However, when it comes to creative expression, the technology still faces inherent limitations in providing low-level controls for localized image stylization. This work enhances state-of-the-art neural style transfer techniques by a generalized user interface with interactive tools to facilitate a creative and localized editing process. Thereby, we first propose a problem characterization representing trade-offs between visual quality, run-time performance, and user control. We then present *MaeSTrO*, a mobile app for orchestration of neural style transfer techniques using iterative, multi-style generative and adaptive neural networks that can be locally controlled by on-screen painting metaphors. At this, first user tests indicate different levels of satisfaction for the implemented techniques and interaction design.**

***Index Terms*—non-photorealistic rendering, style transfer**

I. Introduction

Mobile expressive rendering has become a core technology amongst users that seek casual creativity by image stylization [1], [2] and is continuously supporting the development of mobile artists as a new user group [3]. Image filtering, in particular, takes an essential part of the mobile photo sharing success [4], since filtered photos are more likely to be viewed and commented on by consumers [5]. Image filters are typically implemented by a feature-level engineering approach that enables casual users and mobile artists high-level and low-level interactive control over the stylization process [6], [7], but also limiting it to prescribed stylization effects (e. g., [8]).

A more generalized approach has been introduced by the architecture engineering approach of deep learning, which activates layers of pre-trained deep convolutional neural networks (CNNs) [9] to match content and style statistics, and thus perform a neural style transfer (NST) between arbitrary images [10]. This way, it is able to emulate characteristics of manifold artistic styles and media without deep prior knowledge of photo processing or editing, which is practically demonstrated by mobile applications such as *Prisma* and *PicsArt*. However, in the mobile domain, the technology provided by software products still faces inherent limitations in providing low-level controls for localized image stylization [11]—in contrast to image filtering—e. g., with respect to image feature semantics for meaningful abstraction [12] and support of visual interest [13].

The goal of this work is to implement and enhance state-of-the-art adaptive NST techniques, thereby providing a generalized user interface with creativity tool support [14] for lower-level local control to facilitate the demanding interactive editing [6], [15] on mobile devices (Figure 1). At this, we make the following contributions:

1) We provide a problem characterization with user requirements that are mapped to five functional and non-functional requirements for mobile NST.
2) We present *MaeSTrO*, a mobile app for orchestration of three neural style transfer techniques that can be locally controlled by on-screen painting using image masking.
3) We provide insights from initial user tests, where we report on the levels of satisfaction reached for the implemented techniques and interactive tools.

The remainder of this paper provides a background (Section II), outlines related work (Section III), maps user requirements to functional/non-functional requirements followed

978-1-5386-7315-7/18/$31.00 ©2018 IEEE
DOI 10.1109/CW.2018.00016

* This is the authors' version published as:
Max Reimann, Mandy Klingbeil, Sebastian Pasewaldt, Amir Semmo, Matthias Trapp, Jürgen Döllner, (2018) "MaeSTrO: A Mobile App for Style Transfer Orchestration using Neural Networks", International Conference on Cyberworlds, pp. 9 – 16, DOI: 10.1109/CW.2018.00016

by technical details (Section IV), provides case studies (Section V), and draws conclusions (Section VI).

II. Background

The introduction of mobile expressive rendering had a relevant share in the development of a new user group that discovered the creation of art to be fascinating: *mobile artists*. Their artistic expression is defined through a constant production and discussion of creative imagery, without restriction to a certain skill level or age. As *serious hobbyists* they are always interested in new ways to express their creativity.

We differentiate mobile artists into different groups: Either they start with a blank canvas and their own ideas using low-level drawing or painting techniques, or they use a photograph as input for image processing, e. g., by using the algorithmic support of techniques such as image filtering or example-based rendering via NSTs. We identified the second sub-group as the main target group for *MaeStrO* since they are often early adopters, i. e., they are eager to try new technologies and concepts. To learn more about their requirements, secondary market research, task analysis, user surveys and interviews have been conducted to help design an interactive prototype that fosters creativity, increases productivity and establishes a high user satisfaction when using effects based on NST. Current challenges include the limitation of styles offered in a single app and missing global/local control over an output image, making the usage of such styles unattractive for artists.

a) User requirements: Table I summarizes identified user requirements, i. e., tasks that a system needs to support and the functionalities that need to be provided to support them. Most of these requirements ground on the need of interactive tools required for art direction [16], [17], in particular to widen the interaction spectrum as Isenberg [6] calls for. One of the most important requirements of mobile artists is the *easy exploration of tools and effects*. This includes a self-explanatory interface that enables users to quickly evaluate features, but also a fast image generation process that allows for interactive frame rates (12 frames-per-second or more). Furthermore, *versatile effect presets as well as the possibility to combine different styles* facilitate the development of new ideas. As a limited number of effects also limits creative expression and does not necessarily match a user's stylistic intentions, new ways to explore styles are necessary, e. g., the *possibility to define own styles* through means of experimentation. Beyond that, *artistic control of a style and its parameters* allows for creative expression by customizing and therefore personalizing results. In order to generate unique results, users want to control the stylistic rendering by modifying the configuration parameters locally and globally, as well as by adjusting the visibility, e. g., through blending options. As further modifications with other mobile apps are considered usual, a *high resolution output* is essential for keeping up a high-quality photo editing pipeline.

b) Functional and Non-Functional Requirements: We defined five functional and non-functional requirements of the used NST techniques to meet the expectations of the target group. The *Style Generation Speed (SGS)* defines the overall time that is needed to configure a neural network based on a style image—and a content image—in order to generate an output image via style transfer. The *Style Transfer Speed (STS)* defines the overall time to apply a pre-trained style to a content image to generate a final output image. *Style Transfer Quality (STQ)* refers to the role of filtering in semiotics, i. e., the visual quality with respect to color and features bound to semantics. *User Control (UC)* defines the possibility to alter the stylized output globally or locally by modifying how the style image is used for processing. Finally, *GPU-Memory Consumption (GMS)* describes the required memory of the Graphics Processing Unit (GPU). A higher consumption may technically hinder the application of a NST on mobile devices.

TABLE I: Overview of user requirements mapped to functional and non-functional requirements of NST techniques.

User Requirement	SGS	STS	STQ	UC	GMS
Easy exploration of tools and effects		×	×		
Versatile effects or effect presets	×	×	×	×	
Possibility to define own styles	×	×	×	×	
Artistic control of a style/parameters		×		×	
High resolution output					×

III. Related Work

In the following, we give an overview on NST techniques, approaches for their interactive control and their application in the mobile domain. For comprehensive technical overviews we refer to the work by Jing et al. [18] and Semmo et al. [11].

A. Neural Style Transfer

a) Technical Approaches.: Neural style transfer was first proposed by Gatys et al. [10], who matched global feature statistics in the layers of a deep CNN using Gram matrices, but which is based on a slow and offline optimization process. Several variations have been proposed to this method since then. To enable interactive performance, [19] and [20] trained feed-forward neural networks to directly minimize the same loss as the optimization approach. These networks, however, compromise in flexibility as they are pre-trained with respect to a single style. This problem has been further addressed by custom network layers to match multiple styles: [21] use a conditional layer for instance normalization, [22] use a network with style selection units and [23] embed a CoMatch layer to match second order statistics. These networks can match tens to several hundred styles. Recently, these approaches have then been further extended to arbitrary styles by determining affine parameters for the instance normalization layers at runtime. In particular, [24] compute these parameters directly from the style image in an adaptive instance normalization layer, while [25] use a secondary network to predict them. For *MaeSTrO* we seeked to implement techniques with complementary strengths and weaknesses, thus choosing the optimization-based style transfer of [10], the multi-style feed-forward network of [23] and the adaptive network of [24]. With respect to latter, the approach of [25] may produce results of superior quality but is found to use networks that are too large for a practical usage on mobile devices.

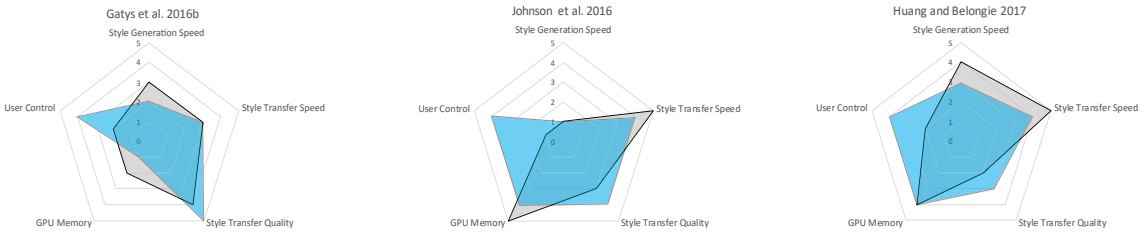


Fig. 2: Subjective comparison of evaluated NST techniques regarding their functional and non-functional requirements (1: worst performance, 5: best performance). Extending existing NST techniques (grey polygon) to facilitate user control over the output leads to higher GPU-memory consumption and a slowed down performance (proposed enhancements: blue polygon).

b) Controllable Neural Style Transfer: Several works proposed methods to improve global and local control over NSTs. Champandard [26] demonstrates the feasibility of a semantic style transfer by using image masks to spatially direct the style application during patch-based transfer [27], an approach that [28] and [29] similarly used for providing style and content masks for Gram-matrix based optimization. Gatys et al. [28] also explored controlling the scale of applied styles, where fine details of a style are separated from coarse structures. While all mentioned methods are developed for iterative NSTs, many are also applicable to feed-forward NSTs. For instance, [23] demonstrate a brush size control for their networks, [24] and [28] show localized stylization control, and [30] use depth information while [31] style/content ratio for control of their respective adaptive feed-forward networks. For *MaeSTrO* we follow [29] for mask-based style guides using iterative NSTs and the approach [24] using adaptive NSTs. Further, we train multi-style feed-forward networks with brush size control [24]. At this, we contribute a loss term for mask-guided transfer in multi-style networks.

B. Mobile Applications

Several mobile apps that utilize NSTs have been created and published on app stores (e. g., for iOS, Android). One of the first apps was *Dreamscope*, which uses an iterative NST approach in a client-server environment, i. e., where a server performs the style transfer. A similar functionality has been implemented by *deepart.io* as a web-based application. Both applications allow to define custom styles via style images, but may require multiple minutes/hours for processing.

Feed-forward NSTs were able to tremendously cut down the processing time [19]. In particular, the photo filtering app *Prisma* was one of the first that successfully used the approach, attracting 30 million users in two months. In its first version, *Prisma* also implemented a client-server-approach, but then opted for an on-device solution for powerful mobile graphics hardware. With the advent of publicly available GPU-based frameworks for execution of neural networks (e. g., CoreML), it is now even feasible to apply mobile feed-forward NSTs in sub-seconds. Popular examples are *Whisky16* and *Pictory* [32] to interactively apply styles to content images, and *DeepStyleCam* [33] to transfer multiple styles in real-time.

However, none of the mentioned apps enable localized control over the style transfer. As a first global control, image-based post processing has been used to define how the stylized output is blended with the content image (e. g., used in *Painnt*). We follow this idea using mask-based painting as an additional constraint for style transfer. The approach is generalized to enable to choose between a quality-based or performance-oriented NST, which we exemplify for the implemented iterative, feed-forward and adaptive techniques.

IV. MaeSTrO

A. Problem Characterization and User Interface

To implement spatial control over the style transfer, the training or configuration of the network can be limited to user-defined regions of the style and content image, as proposed by Luan et al. [29] or Gatys et al. [28]. For *MaeSTrO* we adapted these approaches for all three network types and re-evaluated the most flexible approaches by subjective comparison (Figure 2). Compared to the inflexible approaches, we were able to increase the style transfer quality (Figure 4 and supplemental materials) while causing a higher consumption of GPU memory (GM) through the increased user control to synthesize images, which can be crucial due to the limited memory of mobile devices. In addition, the performance initially slowed down significantly (by factor 2-5, depending on the number of sub-styles), hindering an interactive style transfer. Thus, approaches for interactive modifications with a deferred transfer using multi-style feed-forward and adaptive NSTs are proposed, together with a server-sided iterative NST approach to also synthesize high-quality artistic renditions.

Besides the described technical challenges, the increased conceptual complexity has an extensive impact on usability engineering aspects. Giving local control during style application requires new interaction concepts to achieve a high degree of usability within the system. For the first prototype of *MaeSTrO*, we based our design choices on goals like a high learnability of the system and the reduction of error sources, without losing focus on the user requirements of Section II.

We considered three potential approaches to give artistic control over the output image: (1) Enable users to define and use style brushes, i. e., one style brush is defined through one style image. This allows users to apply multiple styles locally

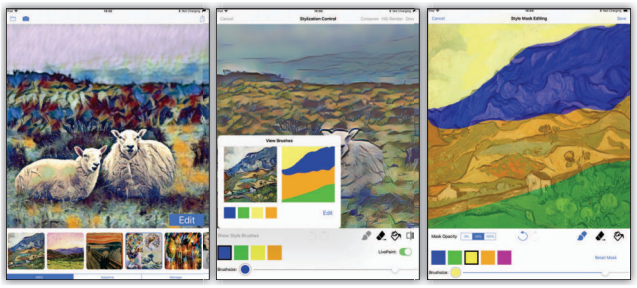


Fig. 3: Overview of important screens in *MaeSTrO*. *Left*: Selection of the NST technique. *Middle*: Editing color-encoded sub-style masks for the adaptive NST. *Right*: Locally applying (sub-)styles to a content image with mask preview. Content image © Rick Barrett on Unsplash.com, used with permission.

to their content image; (2) The usage of detail-controlled style brushes, i. e., applying a style brush with more or less style details through different abstraction levels of the style image; (3) The definition and usage of sub-style brushes.

In our first iOS prototype of *MaeSTrO*, we decided to focus on (3), since this approach addresses the most complex challenges that need to be solved to clearly communicate functionalities to end-users. These include: (a) how to define sub-style brushes as a user, (b) how to visualize pre-defined and user-defined sub-style brushes of a style image and (c) how to apply different sub-style brushes to a content image. Relevant screens of our prototype are shown in Figure 3.

B. Iterative Neural Style Transfer

We adapt the iterative NST of Gatys et al. [10], which defines style transfer as the optimization problem of finding a stylized target image t, whose content is similar to a content image c and whose style is similar to a style image s. At this, a content loss $\mathcal{L}_c$ and a style loss $\mathcal{L}_s$ are minimized using the Gram matrix over activations in a deep neural network. We refer to [10] for formal definitions of the loss terms and optimization. The used Gram matrix sums over the height and width, thus the location of individual features in the style image is lost in the result, and a global texture is transferred.

1) Spatial control and brush size control: Location-based control over NSTs can be achieved by segmenting the style and content image into different local control masks, as described in [29]. The content image is commonly segmented along semantic borders, while masks in the style image are typically chosen to seperate different textures, colors or shapes. To this end, the style loss term is adjusted to include masks

$$\mathcal{L}^l_{s+}(t) = \sum_{m=1}^{C} \| \frac{1}{A^{l,m}(t)} \mathcal{G}^{l,m}(t) - \frac{1}{A^{l,m}(s)} \mathcal{G}^{l,m}(s) \|^2_F \quad (1)$$

$$\mathcal{G}^{l,m}(t) = R\big(\phi^l(t) M^l_m(t)\big) R\big(\phi^l(t) M^l_m(t)\big)^T \quad (2)$$

where C is the number of style and content masks, $\mathcal{G}^{l,m}(\mathbf{t})$ is the Gram matrix of the target for layer l and mask number m, and $M^l_m(t)$ is the content mask m downsampled to the feature map spatial size at layer l, whereas $\mathcal{G}^{l,m}(\mathbf{s})$ conversely yields the Gram matrix for the respective style mask. The area of the masks $A^{l,m}$ is used to normalize magnitude differences in the Gram matrices and reduce intensity artifacts.

In *MaeSTrO*, content masks are either drawn by hand or generated by a semantic segmentation network such as FCN [34]. The generated masks are grouped by the labels into several pre-defined groups based on textural and semantic similarities. The same FCN network is then used as a feature extractor $\phi(t)$ to improve matching between mask and image and to reduce overall memory consumption. The style masks are drawn by hand and brush size control is achieved by varying the size of the style image with the NST [10].

C. Feed-forward Neural Style Transfer

The iterative, spatially controllable approach has a long computation time even for low-resolution images. To speed up the transfer procedure, a feed-forward convolutional neural network, termed the style transfer network, can be trained to learn a direct transformation from content to pastiche [19]. To provide smoothly blended sub-styles, we propose the use of networks capable of capturing multiple styles—instead of training multiple networks—, such as the multi-style generative network (*MSG-Net*) proposed by Zhang and Dana [23], who reduce training time and storage space for multiple styles and enable style blending in feature space.

1) Multi-Style NSTs: The *MSG-Net* [23] learns a generator network $G(c, s)$ that takes both the content and the style targets as inputs, and computes the style Gram matrix at runtime. The main difference to previous work is the introduction of a "CoMatch" layer, which matches second order feature statistics based on the given styles and is able to adapt the output to a set of trainable styles. This layer consists of a learnable weight matrix W, which tunes the content feature map $\phi^l(c)$ based on the target style, where R is a feature vectorization operation:

$$\hat{\phi}^l(c) = R^{-1}\big[R(\phi^l(c))^T W \mathcal{G}(\phi^l(s))\big]^T. \quad (3)$$

The weight matrix W learns to strike a balance between the style and content loss for a particular style.

2) Feature-space mask merging: Similar to [35], we combine different styles in the feature-space, rather than the image-space to prevent artifacts at mask contours and speed up model inference. The *MSG* network conceptually consists of an *encoder* part, which is a siamese network used for feature extraction of both content and style image, a *transformer* part, which carries out the style transformation, and a *decoder*, which is an upsampling convolutional layer to the output image. Training the network in prior works is done without requirements for spatial control, the network is thus optimized towards producing global styles. Using input features with spatially varying styles in the decoder can lead to local image artifacts and certain style elements appearing in differently labeled areas (Figure 4b). We add an additional reguralization term to $\mathcal{L}_{total}(t)$ to diminish this effect (Figure 4c), by adding

(a) Input and style mapping (b) Without $\mathcal{L}_{mask}(t)$ (c) With $\mathcal{L}_{mask}(t)$

Fig. 4: Using the mask loss term $\mathcal{L}_{mask}(t)$ to regularize the decoder for spatially varying styles: (b) *without* masked loss, where dominant style elements (e. g., black mosaic lines) are transferred to differently labeled regions, (c) *with* masked loss, which strictly adheres to the painted style labeling. Content image © Joshua Earle on Unsplash.com, used with permission.

the mean squared error between outputs masked in the feature space and outputs masked in the image space:

$$\mathcal{L}_{mask}(t) = \big\| D\big[\sum_{s\in S} T(t,s)M_s\big] - \sum_{s\in S} G(t,s)M_s \big\|_F^2, \quad (4)$$

where T is the generator network up to the decoder D, $G(t,s) = D(T(t,s))$ is the complete generator network, and M_s the mask for style s. The loss $\mathcal{L}_{mask}(t)$ is added in the last epoch of training, as it is a fine-tuning operation on the decoder network and is computationally more expensive than the other loss terms. To prevent spatial mask-mapping from affecting the encoding and transformation part of the network, the rest of the network is kept fixed during training, except when training with mask-matching (Section IV-C4).

3) Network learning: We follow the procedure and parameters described by Zhang and Dana [23]. In the final epoch, the masked loss $\mathcal{L}_{mask}(t)$ is added to the total loss term $\mathcal{L}_{total}$ and trained using batch size 1 on *COCO-Stuff* [36], a dataset containing dense segmentation masks. We group these masks to the predefined labels and match them according to strategies described in Section IV-C4. We found the *MSG-64* network (64 generator channels) to be sufficient to handle a low number of styles and prefer it because of its superior performance, however it cannot capture a large number of brush sizes or styles compared to *MSG-128*.

4) Learning semantic mappings: Different matching strategies between style and content masks with $\mathcal{L}_{mask}(t)$ during training can have a significant impact on the learned style representations. Style masks are either matched to content masks with a certain label, or to random masks of any class. In the first case, sub-styles are given explicit semantic mappings by manually assigning classes to sub-styles and only matching them with the corresponding masks in the content images, e. g., a sub-style labeled "sky" will only be learned on regions labeled as sky in the training dataset. This method can have superior results for styles with a clear semantic mapping and can be used to enhance existing global NST methods by transferring segments of a style to a content image in a meaningful way, as shown in Figure 5. Using a network trained this way

(a) content/style (b) no matching (c) mask matching

Fig. 5: Networks trained with $\mathcal{L}_{mask}(t)$ to learn a semantic mapping between sub-styles and semantic regions. Content image © karamysh on shutterstock.com, used with permission.

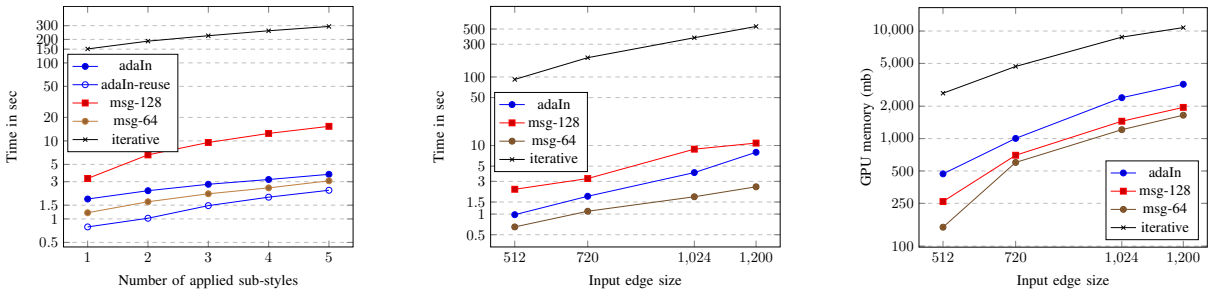
for local control has the drawback that differences between trained semantic mappings and actual semantic content can lead to image artifacts, making this method less flexible. By contrast, randomly matching content masks with styles makes no assumption about the semantic meaning of a style, and is desirable for general-purpose stylization tooling, where regions can be depicted with arbitrary styles.

D. Adaptive Neural Style Transfer

Although *MSG-Net* can learn a large amount of styles, the computational burden for training new styles is still high and therefore does not permit end-users to add custom styles. A recent work by Huang and Belongie [24] introduces a method to perform a NST on arbitrary, new styles by using an encoder-decoder network containing an adaptive instance normalization layer in the middle. The authors use the insight that instance normalization performs a normalization of feature statistics to a certain style using a small set of affine parameters (scale and shift) [37]. They propose an adaptive instance normalization layer, that directly computes these parameters from the first order style feature statistics, i. e., the mean μ and standard deviation σ:

$$\mathit{AdaIN}(c) = \sigma[\phi(s)]\Big(\frac{\phi(c) - \mu[\phi(c)]}{\sigma[\phi(c)]}\Big) + \mu[\phi(s)] \quad (5)$$

where $\phi(s)$ are the style features and $\phi(c)$ the content features. After normalizing the feature maps to the selected style using



(a) #sub-styles vs. run-time (b) image size vs. run-time (c) image size vs. memory consump.

Fig. 6: Performance of the implemented techniques (feed-forward NST tested on an iPad Pro 10.5", iterative NST tested with a NVidia® GTX-1080TI). The time for the adaptive method was measured with and without re-using content and style features.

AdaIn, the decoder transforms the features into the stylized output image. The decoder is trained on a large number of content/style image pairs and learns to reproduce images with arbitrary styles from the style-normalized features.

1) Spatial control and brush size control: To adapt this method to spatial style control, the *AdaIn* normalization of the input to different styles is performed and then jointly decoded [24]. The decoder, although learned on homogeneous inputs, generalizes to inputs of regions with different styles. Both the features of the style and the *AdaIn* feature statistics are computed for each local-control mask at run-time. Normalization to the masked area, similar to the iterative approach, is not required because mean and variance are locally independent.

Brush size control can be achieved by varying the size of the style image input to the encoder, similar to the *MSG-Net* method, but without needing extra training.

E. Implementation

Our implementation consists of three components: (1) A *PyTorch* implementation of the *MSG* model training and style control operations, (2) an implementation targeted to mobile Apple devices based on *iOS* and *CoreML* for on-device image processing and (3) a server-based *PyTorch* implementation of the iterative approach. We extended *CoreML* layers for the *CoMatch* layer of the *MSG* network and the *AdaIn* layer of the adaptive network, for which we provide a CPU-based and a GPU-based implementation using *Metal* shaders. The *MSG CoMatch* layer in the formulation of Equation 3 requires the style features and their Gram matrices to be dynamically computed, but which are compute-intensive operations. To this end, we optimized the performance of the on-device stylization by pre-computing $\hat{W} = \left[W\mathcal{G}(\phi^l(s))\right]^T$ off-device and then computing the *CoMatch* layer output on-device as $\hat{\phi}^l(c) = R^{-1}\left[\hat{W}R(\phi(c))\right]$. The reshaping operation R only reinterprets the memory layout, thus only one on-device matrix multiplication is required for this layer. The performance of the adaptive style transfer network is optimized by caching the computed content and style features (labeled *AdaIn-reuse* in Figure 6), and thus only requiring to re-compute feature means and variances of selected image regions on mask changes. Applying n brushes to one image requires n fast adaptive *InstanceNorm* operations to tune the features to different styles, and one compute-intensive decoder pass.

The performance results in Figure 6 show that increasing the number of applied styles has an almost equal effect on all implemented NST techniques. Increasing the image size is strongly limited by the GPU memory, where sizes of 1200^2 pixels push the limits of available memory on modern mobile devices (typically around 2-4GB) and, for the iterative method, available memory on high-end desktop GPUs (typically 11-16GB). Producing very high-resolution images is therefore not possible with the given tools and would require either new, smaller network architectures, or image tiling procedures.

V. Case Studies and User Tests

MaeSTrO combines three NST techniques in a single app, thus enabling users to create cross-stylized outputs.

A. Case Studies

The comparisons in Figure 7 indicate that the NST techniques yield visually different outputs. In particular, the iterative approach of [10] and the adaptive approach of [24] largely fail to transfer color features. By contrast, our enhancements enable local control over the NST, which facilitates the implementation of a semantic mapping of a style image to a content image, i. e., currently by explicit manual labeling of image regions—as an enhancement this could also be performed using GIST descriptors [38], [39]. This leads to a more plausible transfer of texture and color features for the iterative and feed-forward approach. Although the adaptive approach generally lacks to produce a style-like result, a clear separation between the different sub-styles could be achieved.

Our approaches have in common that sub-styles are mapped to masked areas of the content image. This enables different application scenarios ranging from a semantics-based, over a separated fore- and background to a completely free stylization. Although the mask creation process can be automated, we find that the manual approach yields better results (Figure 7). This is especially true for regions with sparse color or texture features, which may lead to miss-segmentations. Nevertheless, an automatic segmentation can be useful as a starting point for casual users, since it eases the refinement of the masks.

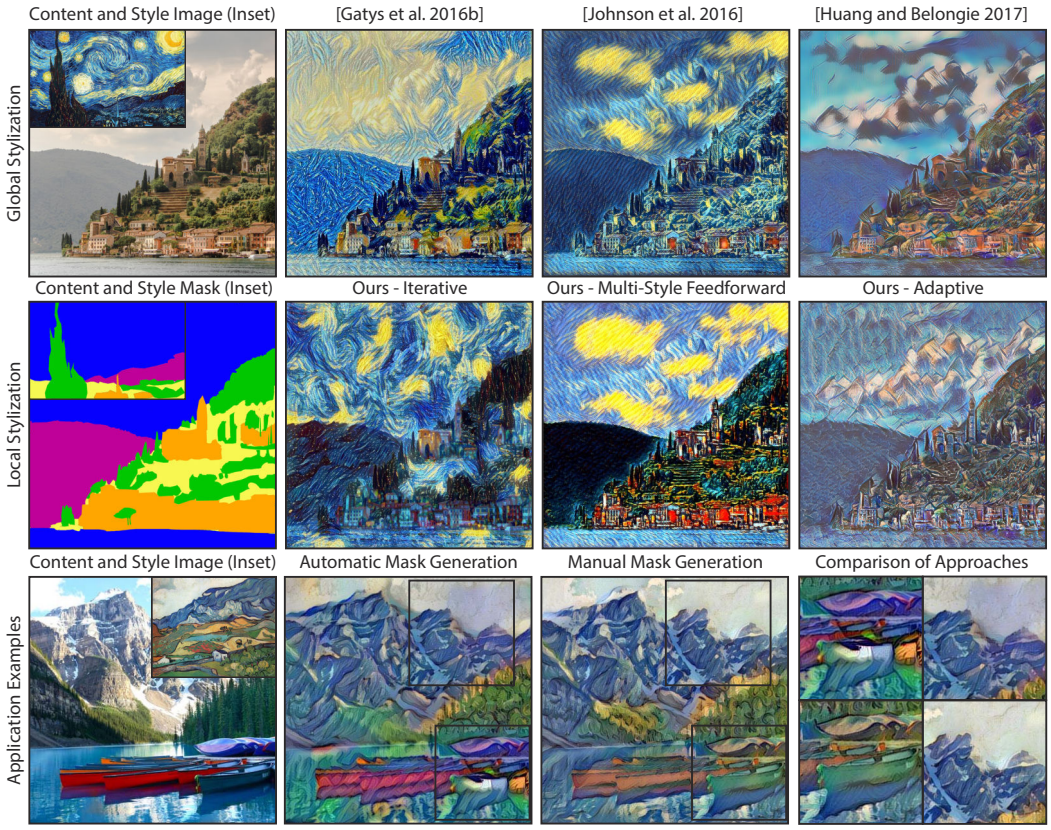


Fig. 7: Outputs created with *MaeSTrO*. Local control enables semantically more plausible results and clearer separations between image regions. Content images © Matthew Fournier and Henrique Ferreira on Unsplash.com, used with permission.

B. User Tests

A first usability study has been conducted with six participants using the prototype on an Apple iPad Pro 12,9" (1st generation). The participants are mainly casual users, between 16 and 37 years old and have mixed experiences with photo manipulation apps. We asked them to perform specific tasks to test the design choices we made. As an example, test users were asked to globally apply the style of "The Starry Night" to a prepared content image using the multi-style generative approach. As already shown in Figure 7, a global stylization would create semantically incorrect results, e. g., by not producing a starry sky, but instead colorizing the foreground into yellow. Thus, users should make specific local modifications with the pre-defined sub-style brushes.

All users were convinced that the local control improved the overall style quality. However, we could observe several challenges when learning the functionalities of the app and utilizing them for their purposes. First, the global application of a style onto a content image caused misleading expectations and hindered the creation of content masks. Two users expected that the app would automatically create content masks based on the semantics used within the style masks. Other speculations included that the app would give recommendations for the application of defined sub-style brushes to sub-regions of the content image. The majority of testers (five out of six) also missed a direct comparison between the globally stylized image and the image with their local changes. Four would have preferred to work directly on the original content image.

Second, the mapping of sub-style brushes through colors has been misleading when working with images, as it would directly lead to the idea of "colorizing the picture". We experimented with two different approaches: Letting users define a style and a style mask by themselves first, or let them directly edit a content image locally by giving them a style with pre-defined sub-styles. For the first approach, none of the user testers could identify what they needed to do within an acceptable amount of time ($<$ 5 minutes). In the second approach, an overlay with a small preview of the style image and the style mask helped users to realize the purpose of the colors. In general, after working on several content images within the app, five out of six users wished not to use colors for mapping purposes, but semantically meaningful symbols or thumbnails with another kind of highlighting for the corresponding regions.

The issue of mapping style and content masks overlaps with a third challenge: the fast exploration of local changes in a content image. This only applies to multi-style generative and adaptive neural networks, as the iterative approach is server-

based and does not allow a preview of the sub-style that should be locally applied. When editing a content image, users were able to view the content mask, showing the applied colors, or the low-rendered preview with insights about the expected results. Four out of six users prefer the preview, but switched up to 8 times per minute to produce satisfying results. They used the content mask to make sure they did not miss to mask any parts of an edited region, and switched back to the preview to make relevant changes on their artwork.

VI. Conclusions

In this work, we implement and extend three techniques for neural style transfer with local control on mobile devices. We show that iterative, multi-style feed-forward and adaptive style transfers are complementary in terms of run-time, user control and visual quality, and conclude that offering all three in a single, end-user oriented mobile app achieves a high degree of flexibility for creating expressive results. We introduce an UI concept which mixes interactive brush-based stylizations with deferred, high-quality stylizations. Evaluating the mobile app on casual users shows that local control methods provide tangible benefits to this user group over global transfer, such as increased artistic freedom and superior stylization quality, even with little to no training.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback. This work was funded by the Federal Ministry of Education and Research (BMBF), Germany, for the AVA project 01IS15041.

References

[1] K. Dev, "Mobile Expressive Renderings: The State of the Art," *IEEE Computer Graphics and Applications*, vol. 33, no. 3, pp. 22–31, 2013.

[2] H. Winnemöller, "NPR in the Wild," in *Image and Video based Artistic Stylisation*, ser. Computational Imaging and Vision, P. Rosin and J. Collomosse, Eds. London/Heidelberg: Springer, 2013, vol. 42, ch. 17, pp. 353–374.

[3] D. Keep, *Artist with a Camera-Phone: A Decade of Mobile Photography*. New York: Palgrave Macmillan US, 2014, pp. 14–24.

[4] M. Berry, "Re–imagining Place with Filters: More Than Meets the Eye," *Journal of Creative Technologies*, vol. 4, 2014.

[5] S. Bakhshi, D. A. Shamma, L. Kennedy, and E. Gilbert, "Why We Filter Our Photos and How It Impacts Engagement," in *Proc. ICWSM*, 2015, pp. 12–21.

[6] T. Isenberg, "Interactive NPAR: What Type of Tools Should We Create?" in *Proc. NPAR*. Goslar, Germany: Eurographics Association, 2016, pp. 89–96.

[7] A. Semmo, T. Dürschmid, M. Trapp, M. Klingbeil, J. Döllner, and S. Pasewaldt, "Interactive Image Filtering with Multiple Levels-of-control on Mobile Devices," in *Proc. MGIA*. New York: ACM, 2016, pp. 2:1–2:8.

[8] M. Klingbeil, S. Pasewaldt, A. Semmo, and J. Döllner, "Challenges in User Experience Design of Image Filtering Apps," in *Proc. MGIA*. New York, NY, USA: ACM, 2017, pp. 22:1–22:6.

[9] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv.org report 1409.1556, Apr. 2015.

[10] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," in *Proc. CVPR*. Los Alamitos: IEEE Computer Society, 2016, pp. 2414–2423.

[11] A. Semmo, T. Isenberg, and J. Döllner, "Neural Style Transfer: A Paradigm Shift for Image-based Artistic Rendering?" in *Proc. NPAR*. New York: ACM, 7 2017, pp. 5:1–5:13.

[12] D. DeCarlo and A. Santella, "Stylization and Abstraction of Photographs," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 769–776, Jul. 2002.

[13] A. Santella and D. DeCarlo, "Visual Interest and NPR: An Evaluation and Manifesto," in *Proc. NPAR*. New York, NY, USA: ACM, 2004, pp. 71–150.

[14] B. Shneiderman, "Creativity Support Tools: Accelerating Discovery and Innovation," *Commun. ACM*, vol. 50, no. 12, pp. 20–32, Dec. 2007.

[15] J. Seims, "Putting the artist in the loop," *ACM SIGGRAPH Computer Graphics*, vol. 33, no. 1, pp. 52–53, 1999.

[16] D. H. Salesin, "Non-Photorealistic Animation & Rendering: 7 Grand Challenges," Keynote talk at NPAR, Jun. 2002.

[17] A. A. Gooch, J. Long, L. Ji, A. Estey, and B. S. Gooch, "Viewing Progress in Non-Photorealistic Rendering Through Heinlein's Lens," in *Proc. NPAR*. New York: ACM, 2010, pp. 165–171.

[18] Y. Jing, Y. Yang, Z. Feng, J. Ye, and M. Song, "Neural Style Transfer: A Review," arXiv, Tech. Rep. abs/1705.04058, 2018.

[19] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," in *Proc. ECCV*. Cham, Switzerland: Springer International, 2016, pp. 694–711.

[20] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky, "Texture Networks: Feed-forward Synthesis of Textures and Stylized Images," in *Proc. International Conference on Machine Learning*. New York: JMLR.org, 2016, pp. 1349–1357.

[21] V. Dumoulin, J. Shlens, and M. Kudlur, "A Learned Representation For Artistic Style," arXiv.org report 1610.07629, Feb. 2017.

[22] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Diversified Texture Synthesis with Feed-Forward Networks," in *Proc. CVPR*, 2017, pp. 266–274.

[23] H. Zhang and K. Dana, "Multi-style Generative Network for Real-time Transfer," arXiv.org report 1703.06953, Mar. 2017.

[24] X. Huang and S. Belongie, "Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization," arXiv.org report 1703.06868, Mar. 2017.

[25] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens, "Exploring the structure of a real-time, arbitrary neural artistic stylization network," arXiv.org report, May 2017.

[26] A. J. Champandard, "Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artworks," arXiv.org report 1612.04337, Mar. 2016.

[27] C. Li and M. Wand, "Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis," in *Proc. CVPR*. Los Alamitos: IEEE Computer Society, 2016, pp. 2479–2486.

[28] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman, "Controlling Perceptual Factors in Neural Style Transfer," in *Proc. CVPR*. Los Alamitos: IEEE Computer Society, July 2017, pp. 3730–3738.

[29] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep Photo Style Transfer," arXiv.org report abs/1703.07511, 2017.

[30] X.-C. Liu, M.-M. Cheng, Y.-K. Lai, and P. L. Rosin, "Depth-aware Neural Style Transfer," in *Proc. NPAR*, 2017, pp. 4:1–4:10.

[31] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Universal style transfer via feature transforms," in *Advances in Neural Information Processing Systems*, 2017, pp. 385–395.

[32] A. Semmo, M. Trapp, J. Döllner, and M. Klingbeil, "Pictory: Combining Neural Style Transfer and Image Filtering," in *Proc. SIGGRAPH Appy Hour*. New York, NY, USA: ACM, 2017, pp. 5:1–5:2.

[33] R. Tanno, S. Matsuo, W. Shimoda, and K. Yanai, "DeepStyleCam: A Real-Time Style Transfer App on iOS," in *Proc. MultiMedia Modeling*. Cham, Switzerland: Springer International, 2017, pp. 446–449.

[34] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," arXiv.org report abs/1605.06211, 2016.

[35] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, "StyleBank: An Explicit Representation for Neural Image Style Transfer," in *Proc. CVPR*. Los Alamitos: IEEE Computer Society, 2017, pp. 2770–2779.

[36] H. Caesar, J. R. R. Uijlings, and V. Ferrari, "COCO-Stuff: Thing and Stuff Classes in Context," arXiv.org report abs/1612.03716, 2016.

[37] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance Normalization: The Missing Ingredient for Fast Stylization," arXiv.org report 1607.08022, Sep. 2016.

[38] M. Toyoura, N. Abe, and X. Mao, "Painterly Image Generation Using Scene-Aware Style Transferring," in *Proc. International Conference on Cyberworlds*. Los Alamitos: IEEE Computer Society, 2016, pp. 73–80.

[39] ——, *Scene-Aware Style Transferring Using GIST*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 29–49.